understood that not all meta-implementation layers of the present invention include all of the types of descriptors and implementations of the meta-implementation layer shown in **FIG. 1**

[0112] An enumeration descriptor describes a set of named constants. Enumeration descriptors assign a string value to the name that is used in the implementation to retrieve the value for that name. The enumeration descriptor associates this name with a configuration descriptor that holds the attribute values necessary to create an instance of that value. Enumeration descriptors are a datatype used by attribute descriptors and parameter descriptors when the values expected fall into a limited set of known constants.

[0113] "Other element descriptors" or "future element descriptors" may be added to the metamodel to represent new computing technology concepts not already captured in other elements in the meta-implementation layer. These future elements are first be defined in the meta-metarepository to define their structure and use. After being defined in the meta-metamodel repository, instances of these descriptors can be added to the metamodel repository.

[0114] An enumeration implementation is a set of constant named values. The enumeration holds the name value pairs and can retrieve the value for a given name and can retrieve the name for a value. The names and values are constant and cannot be changed once added to the enumeration.

[0115] Other element implementations or "future element implementations" are defined for each "other element descriptor". These future implementation types will define the features and functionality necessary for a future technology or future computer concept to be added to the meta-implementation layer. The element implementation will provide a mechanism to create instances of itself (as is required of all implementations) as well as a mechanism to use the implementation or instances for the purpose the element implementation fulfills.

[0116] **FIG. 2** illustrates metametamodel repository of **FIG. 1** in greater detail. The metametamodel repository includes metamodel descriptors and metamodel implementations (metamodel descriptors describe model descriptors and metamodel implementations describe model implementations). The Meta-metamodel repository includes metaenumeration descriptors, metarole descriptors, metahint descriptors, metadatatype descriptors, metaconstraint descriptors, metaattribute descriptors, other meta-element descriptors, metaparameter descriptors, metamethod descriptors, metasignal descriptors, metainterface descriptors, metamodel descriptors, and metapackage descriptors. Implementations include includes metaenumeration implementations, metarole descriptors, metahint implementations, metadatatype implementations, metaconstraint implementations, metaattribute implementations, other meta-element implementations, metaparameter implementations, metamethod implementations, metasignal implementations, metainterface implementations, metamodel implementations, and metapackage implementations.

[0117] Each of the types of metamodel descriptors of the metametamodel repository of **FIG. 2** is associated with a similarly name descriptor in the metamodel repository. For example, the metaenumeration descriptors of the metamodel descriptors are associated with the enumeration descriptors of the metamodel repository. Similarly each of the types of metamodel implementation is associated with similarly named implementations. For example, the metaenumeration implementations are associated with the enumeration implementations. Also, it should be understood that not all metametamodel repositories of the present invention include all of the types of metamodel descriptors and metamodel implementations of the metamodel repository shown in **FIG. 2**.

[0118] **FIG. 3** illustrates a component integration engine of the present invention that includes the meta-implementations layer of **FIG. 1** and instances. The instances include enumeration instances, role instances, hint instances, datatype instances, constraint instances, attribute instances, other element instances, parameter instances, method instances, signal instances, interface instances, model instances, and package instances. The component integration engine also includes shared services, pluggable authentication, component assemblies, persistence engines, flow chart assemblies and user access points.

[0119] Shared services are a set of operations used in software applications that are available as part of the component integration engine. These services may provide access to platform specific functionality (like hardware access, network listening, file storage, etc.), performance sensitive functionality (like real-time scheduling, mathematical computations, etc.) or very common functionality (like database access, email access, formatting, message queuing, object caching, etc.). A shared service must implement the service interface that specifies how to start, stop, and pause the service. Otherwise, shared services are integrated into the component integration engine exactly like any other component; they are described by descriptors and accessible through the meta-implementation layer. The only shared service required in a component integration engine is a service to facilitate the control of the component integration engine itself. This service provides the support for configuring the component integration engine and starting, stopping, or pausing the component integration engine.

[0120] The pluggable authentication layer provides a standardized interface for authenticating users of the component integration engine and assigning the appropriate credentials to those users. Authentication is the process of verifying a user is who the user claims to be. Authorization is the process of granting permission to a user. Authorization credentials are generally assigned during the authentication process and are generally checked by access constraints. Credentials may be created and assigned through other means and authorization may be enforced by other mechanisms. The pluggable authentication layer allows different authentication components to be "plugged-in" or switched around to provide the best component for performing the authentication.

[0121] Component assemblies are groups of components with defined relationships used together to perform a specific task. A component assembly is similar to a model instance. Both component assemblies and model instances have a set of data instances (models call these attributes, component assemblies call these members). Both component assemblies and model instances have a set of operations that can be performed upon them (models call these methods, component assemblies call these operations). The dif-